Analysis Report on Inspecting MQTT

Liyao Tang - u6142160

May 21, 2018

1 Analysis on Handshakes under Different QoS

Figure 1 shows the required screenshots. QoS defines the handshake of the sending and receiving of one message between the sender and receiver, which both can be either broker or devices. For each QoS level, an explanation of its handshake are given as follow.

1. QoS = 0

   The application message is delivered according to the best efforts of the underlying TCP/IP network and sender would discard the application message once sent out.

   Hence, the application message will arrive at receiver at most once.

2. QoS = 1

   After application message sent, sender waits for an acknowledgement (PUBACK or Publish Ack) to make sure the application message is received. To match PUBACK with corresponding application message, each application message at this QoS level has an ID. After PUBACK received, application message can be safely discarded.

   After a predefined time without returning PUBACK for the application message, sender will re-send application message with its DUP flag set, meaning this is duplicate.

   Hence, the application message will arrive at receiver at least once.

3. QoS = 2

   After application message sent, sender waits for an acknowledgement (PUBREC or Publish Received) to ensure application message is received and then responds with a further acknowledgement (PUBREL or Publish Release) to acknowledge that it knows application message is received. Afterwards, sender still needs to wait for one more acknowledgement (PUBCOMP or Publish Complete) from receiver so that sender is sure that receiver has received the PUBREL. Finally, the application message can be safely discarded.

   After a predefined time without the expecting message, the protocol (either sender or receiver) will retry from the last unacknowledged message.

   Hence, the application message will arrive at receiver exactly once.

No.	Time	Source	Destination	Protoc	Length	Info
4	0.077017…	150.203.213.195	52.65.194.50	MQTT	118	Connect Command
6	0.158035…	52.65.194.50	150.203.213.195	MQTT	72	Connect Ack
8	0.158642…	150.203.213.195	52.65.194.50	MQTT	90	Subscribe Request
10	0.324578…	52.65.194.50	150.203.213.195	MQTT	73	Subscribe Ack
12	0.405582…	52.65.194.50	150.203.213.195	MQTT	92	Publish Message
14	0.982773…	52.65.194.50	150.203.213.195	MQTT	90	Publish Message
16	1.966636…	52.65.194.50	150.203.213.195	MQTT	90	Publish Message
18	1.967216…	150.203.213.195	52.65.194.50	MQTT	70	Disconnect Req

(a) QoS = 0

No.	Time	Source	Destination	Protoc	Length	Info
4	0.035038…	150.203.213.195	52.65.194.50	MQTT	118	Connect Command
6	0.116371…	52.65.194.50	150.203.213.195	MQTT	72	Connect Ack
8	0.117783…	150.203.213.195	52.65.194.50	MQTT	90	Subscribe Request
9	0.197431…	52.65.194.50	150.203.213.195	MQTT	73	Subscribe Ack
11	0.278441…	52.65.194.50	150.203.213.195	MQTT	94	Publish Message
13	0.279013…	150.203.213.195	52.65.194.50	MQTT	72	Publish Ack
15	0.855686…	52.65.194.50	150.203.213.195	MQTT	92	Publish Message
16	0.855944…	150.203.213.195	52.65.194.50	MQTT	72	Publish Ack
18	1.918678…	52.65.194.50	150.203.213.195	MQTT	92	Publish Message
19	1.919294…	150.203.213.195	52.65.194.50	MQTT	72	Publish Ack
20	1.919525…	150.203.213.195	52.65.194.50	MQTT	70	Disconnect Req

(b) QoS = 1

No.	Time	Source	Destination	Protoc	Length	Info
4	0.012693…	150.203.213.195	52.65.194.50	MQTT	118	Connect Command
6	0.093942…	52.65.194.50	150.203.213.195	MQTT	72	Connect Ack
8	0.094997…	150.203.213.195	52.65.194.50	MQTT	90	Subscribe Request
9	0.175283…	52.65.194.50	150.203.213.195	MQTT	73	Subscribe Ack
11	0.260731…	52.65.194.50	150.203.213.195	MQTT	94	Publish Message
13	0.261244…	150.203.213.195	52.65.194.50	MQTT	72	Publish Received
14	0.341834…	52.65.194.50	150.203.213.195	MQTT	72	Publish Release
15	0.342358…	150.203.213.195	52.65.194.50	MQTT	72	Publish Complete
17	0.996354…	52.65.194.50	150.203.213.195	MQTT	92	Publish Message
18	0.996784…	150.203.213.195	52.65.194.50	MQTT	72	Publish Received
20	1.076962…	52.65.194.50	150.203.213.195	MQTT	72	Publish Release
21	1.077539…	150.203.213.195	52.65.194.50	MQTT	72	Publish Complete
23	2.064348…	52.65.194.50	150.203.213.195	MQTT	92	Publish Message
24	2.064815…	150.203.213.195	52.65.194.50	MQTT	72	Publish Received
26	2.145230…	52.65.194.50	150.203.213.195	MQTT	72	Publish Release
27	2.145740…	150.203.213.195	52.65.194.50	MQTT	70	Disconnect Req

(c) QoS = 2

Figure 1: Figures of MQTT handshake under different QoS. The client disconnects the broker after receives three messages.

As listed above, those assisting messages that is not compulsory for application to meet its specification can transmit on QoS level 0; whereas those key messages, which may results in deviation from the specification if not received, should transmit under QoS level 1; while some important messages whose not only payload but also occurrence matter should transmit under QoS level 2 because duplicate messages are not applicable under this circumstance.

2 Statistical Analysis for Each QoS Level

2.1 Collecting Statistics

2.1.1 Code Structure and Explanation

The code is written in python3.6.4, using MQTT library and several other standard python libraries, including OptionParser, time, sys and gc.

The code takes two compulsory command line parameters, explained as follow:

- "- -speed" It takes one of following values: ("fast", "slow", "SYS"), where "fast" stands for the fast channel, "slow" for the slow channel and "SYS" for the "$SYS/broker/#" channel.
- "- -qos" It takes one of following values: (0, 1, 2), which corresponds to the MQTT qos level of. If the "–speed" takes "SYS", then "–qos" must be 2.

If the script subscribes to one of the fast channels, it will collect the statics for 1 minutes and record it into a local log and will publish the required statics after 10 minutes. Then it will starts another round of collecting statics, recording logs and publishing result. All the recorded time are local time-stamp.

If the script subscribes to the "$SYS/broker" topic, it will write whatever it receives into a local log file with a local time-stamp.

In practice, four scripts will run in parallel, outputting into separate files, referring to the local time on the same computer when time-stamping messages. It is shown that the script is able to run for days with a nearly constant memory consumption on a PC.

2.1.2 Statics Definition

As we are requested multiple statics, some special name will be used in the following report are first introduced and then it will be given the concrete definition and calculation for those statics and some other statics collected in my code for the purpose of analysis.

Special name:

- valid message If the client subscribes to one of the fast channel then a valid message is any message from the topic whose payload can be directly translated into an integer.

- expected number If the client subscribes to one of the fast channel and the latest number in received valid messages is x then either $x+1$ ot 0 is the expected number. Yet, if the client subsribes to one of the fast channel and has not received any number from a valid message from the topic, than any integer is expected.

- ordered message If the client subscribes to one of the fast channel and the latest time stamp in received valid messages is t, than ordered message is a valid message with a time stamp t' such that $t' > t$. Yet, if the client subsribes to one of the fast channel and has not received any valid message from the topic, than any valid message is ordered.

- expected message If the client subscribes to one of the fast channel then the expected message is a ordered message containing a expected number.

- duplicate message The duplicate messages are considered only under qos 1 and 2 because the broker by definition should not re-send any message under qos level 0.

  Under qos level 1 or 2, if the client subscribes to one of the fast channel, duplicate message is any ordered message with the same number as the current one. (An ordered message with a number bigger or smaller than the current one are considered an loss in messages that are, correspondingly, either before or after wraps around.)

- mis-ordered message

  If the client subscribes to one of the fast channel then any message that is not considered ordered message is a mis-ordered message.

Definition of statics:

- 1-min expected count

  The expected count are the total number of expected message within the 1-minute interval.

- 1-min rate of messages received

  This statics are the number of valid messages received in the 1-minute interval.

- 1-minute loss rate

  This statics are calculated by following equation 1.

$$\text{loss rate} = \frac{losscount}{count} \tag{1}$$

  where *count* is the 1-min rate of messages received and *losscount* are the lost messages within this 1-minute interval. The messages loss are recognised whenever it is received an ordered message with a number not consistent with the expected number. The worst 1-minute loss rate is the biggest loss rate encountered in 10-minute interval.

- worst 1-minute duplicate rate

  Similar to 1-minute loss rate, it is calculated by equation 2

$$\text{loss rate} = \frac{dupecount}{count} \tag{2}$$

  where *count* is the 1-min rate of messages received and *dupecount* are the duplicate messages with in this 1-minute interval. Worst 1-minute duplicate rate is then the biggest 1-minute loss rate encountered in the 10-minute interval.

- 1-minute out-of-order rate

  out-of-order rate is calculated by equation 3

$$out - of - orderrate = \frac{mis - ordercount}{count} \tag{3}$$

  where *count* is the 1-min rate of messages received and $mis-ordercount$ is the number of mis-ordered message encountered in the 1-minute interval. Worst 1-minute out-of-order rate is then the biggest 1-minute out-of-order rate encountered in the 10-minute interval.

2.2 Correlation with $SYS

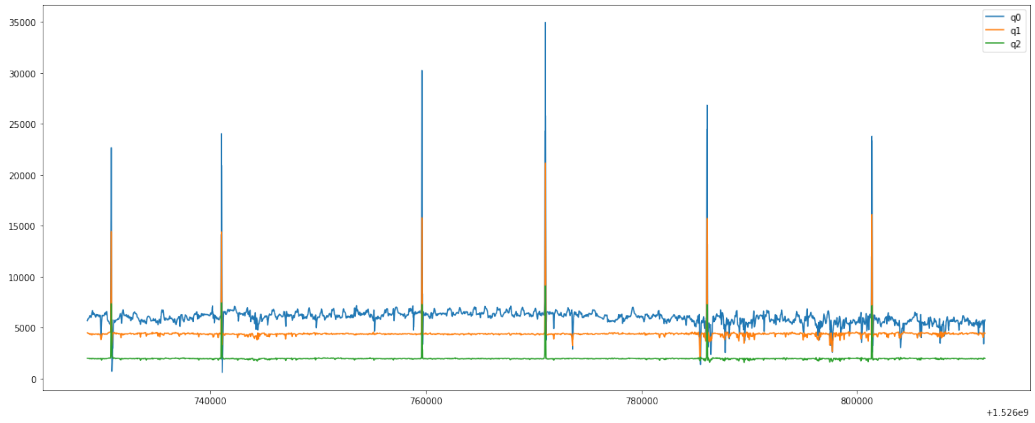
Besides the required fields, there are other topics analysed in the following analysis and they are labelled correspondingly in their graphs. The following graph are the time span from 2018-05-19 21:17:50 to 2018-05-20 20:22:14.

Convention of plotting in this section is that the information collected by clients are plotted in either blue, orange or green while the SYS statics are in red. The related discussions are placed under each graph and cross-referenced.

2.2.1 Expected Messages Correlating with $SYS

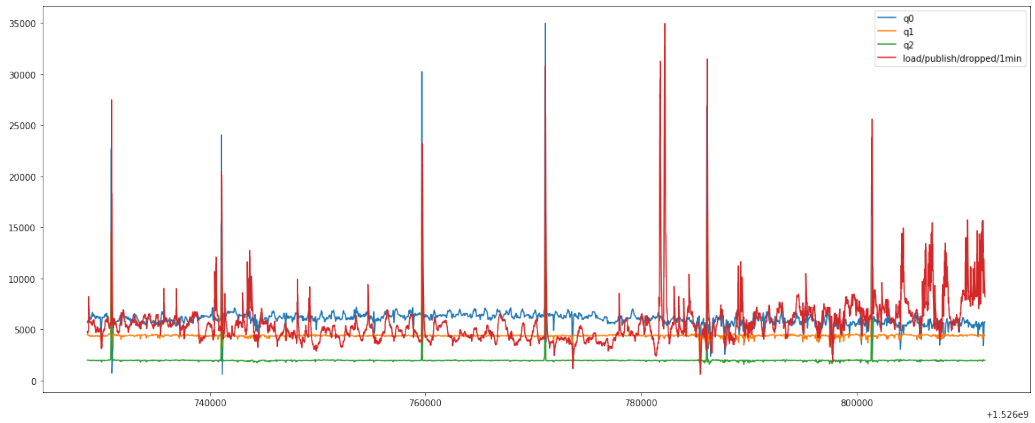
To measure the performance of the broker along the time series, the number of expected messages over 1 minute under each QoS level are plotted into graphs. In order to show the correlation with SYS statics, the SYS statics are re-scaled appropriately so that the trends and details of both number of expected messages and the SYS statics can be expressed.

Figure 2: 1-min Expected Messages under QoS 0,1,2



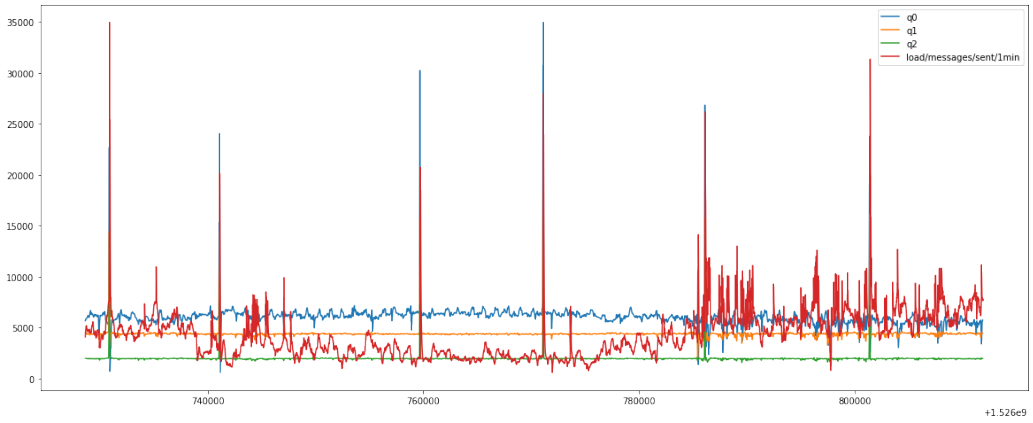
As convention, the blue, orange and green curves are 1-min expected messages number under QoS level respectively 0, 1 and 2.

Figure 3: topic: load/publish/dropped/1min



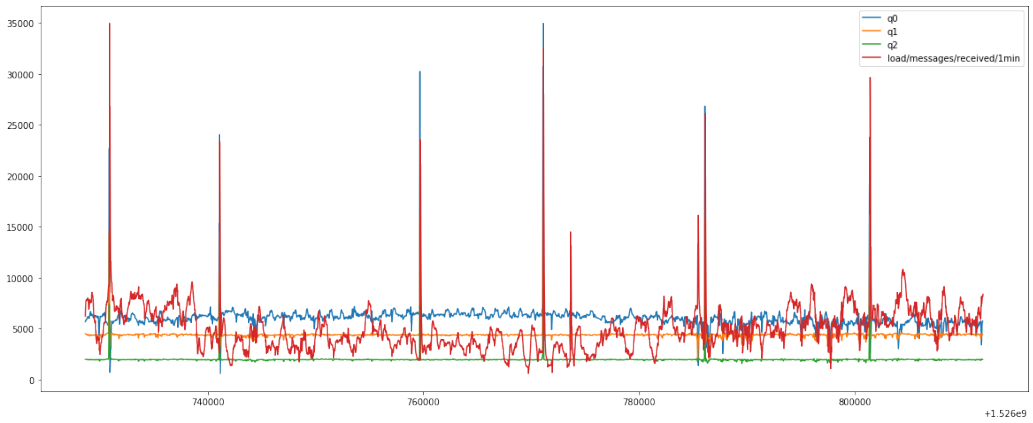
It is observed that, surprisingly, the dropped messages largely correlates with the number of received expected messages, except for the fifth; it yet is reasonable because each peak is actually a publishing peak (shown in next figure 4), when the message dropping is severe as expected due to the jam.

Figure 4: topic: load/messages/sent/1min



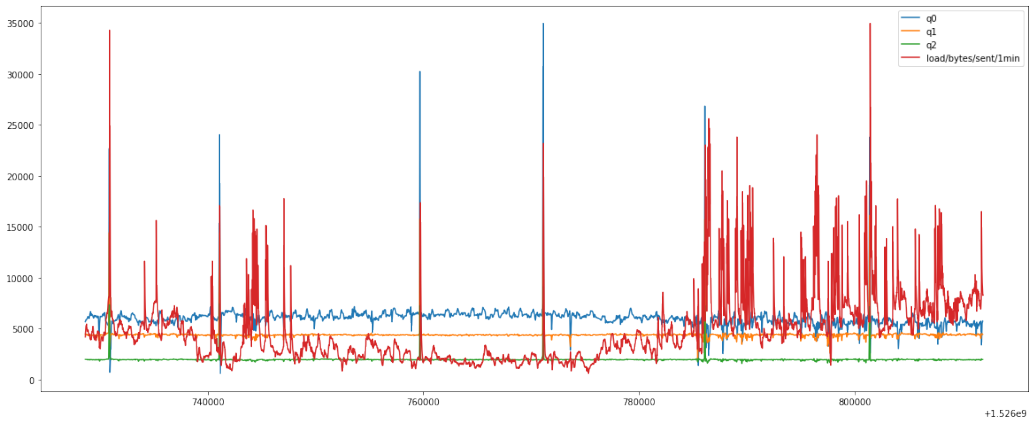
All the peak of 1-min expected messages number for each QoS nicely correlates with the sending peak from the SYS statics, which is expected because more it sends, more clients receive, and more chance for client to have an expected sequential stream of received number.

Figure 5: topic: load/messages/received/1min



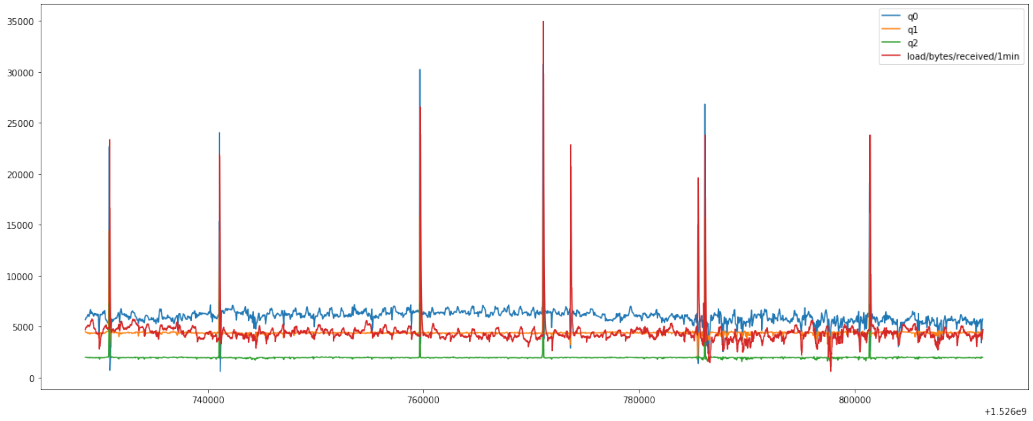
Again, all the peaks from SYS statics, except for some small peaks, correlates with curves for expected messages. This is expected because messages received by clients fundamentally come from the fast counter and broker needs to receives those numbers before passing them to clients.

Figure 6: topic: load/bytes/sent/1min



This is actually an interesting plot because it accounts for the size of each messages, where we can observe that on the left hand side around the first x-tick (740000) and after the third x-tick (780000), there are a number of small peaks of sending, denoting peaks of broker's work load for sending messages, which nicely correlates with those jitter-like peaks in previous figures (load/message/sent/1min: fig 4, load/publish/dropped/1min: fig 3)

Figure 7: topic: load/bytes/received/1min



It is however interesting that the messages received by the broker is not consistent with those outgoing peaks. Then why would broker suddenly needs to send bigger and more messages?

Figure 8: topic: load/connections/1min

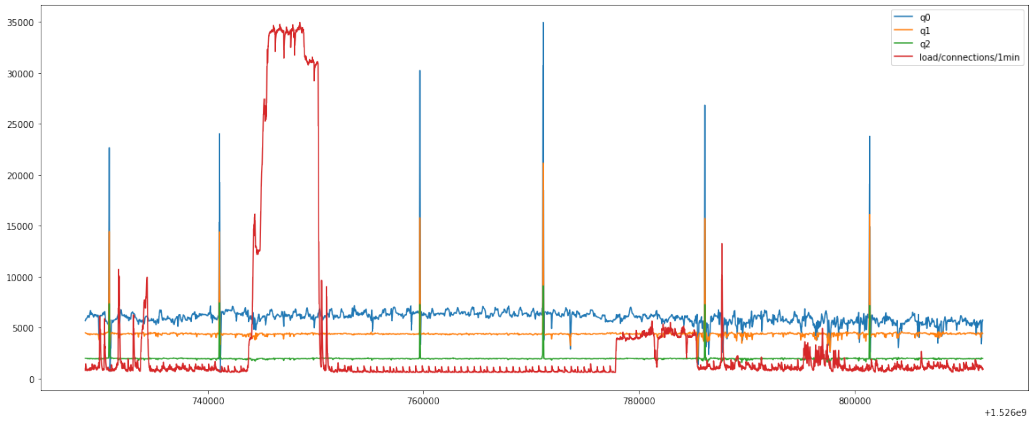


Figure 9: topic: clients/connected

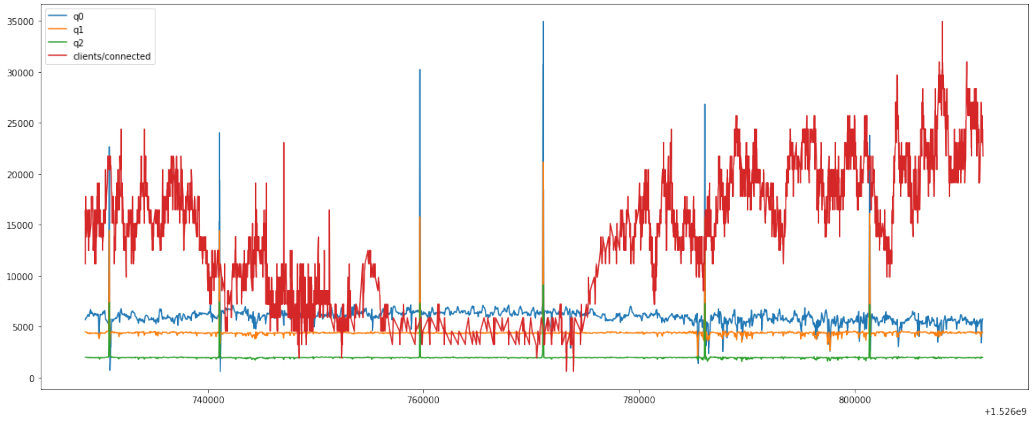


Figure 10: topic: heap/current

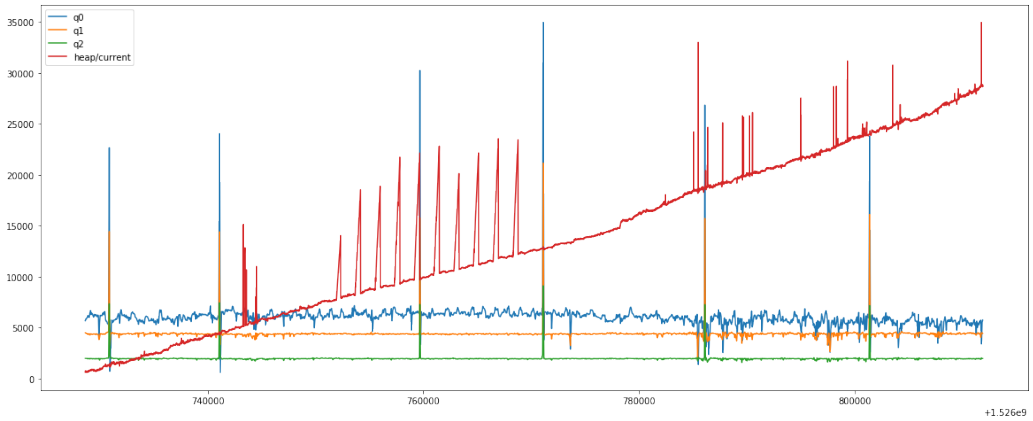


Figure 11: topic: heap/maximum

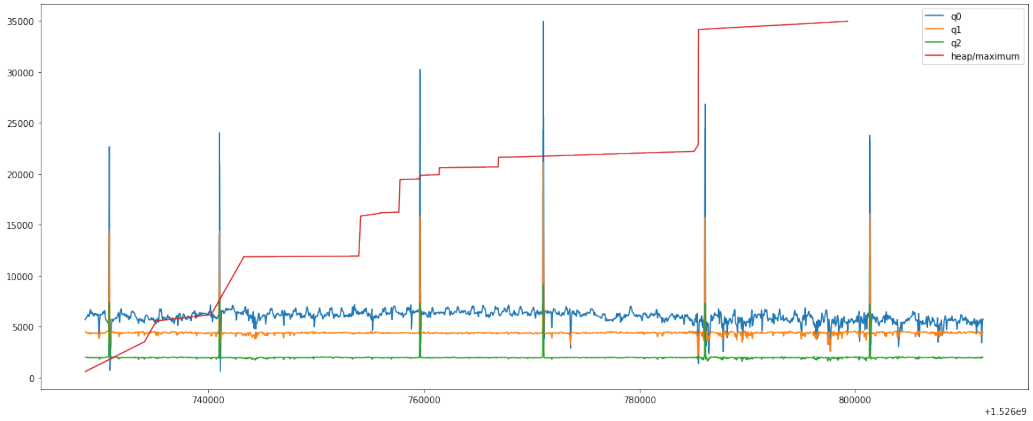


Figure 12: topic: messages/stored

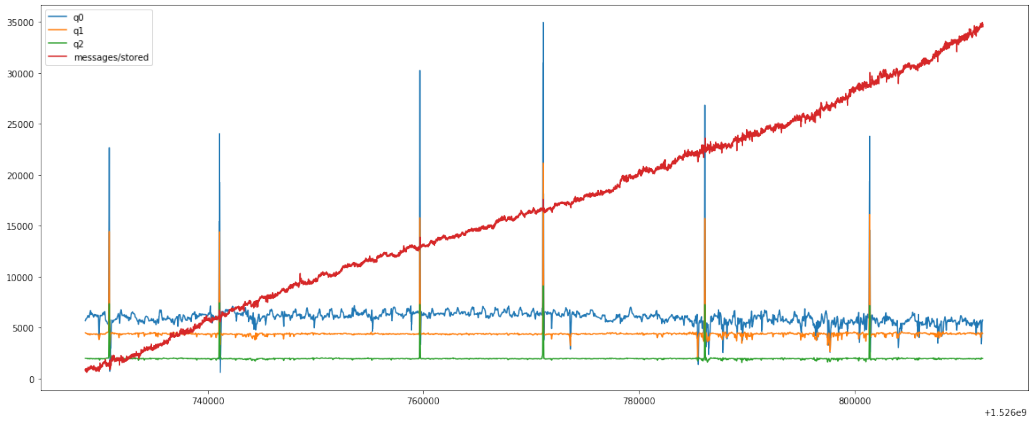


Figure 13: topic: publish/messages/dropped

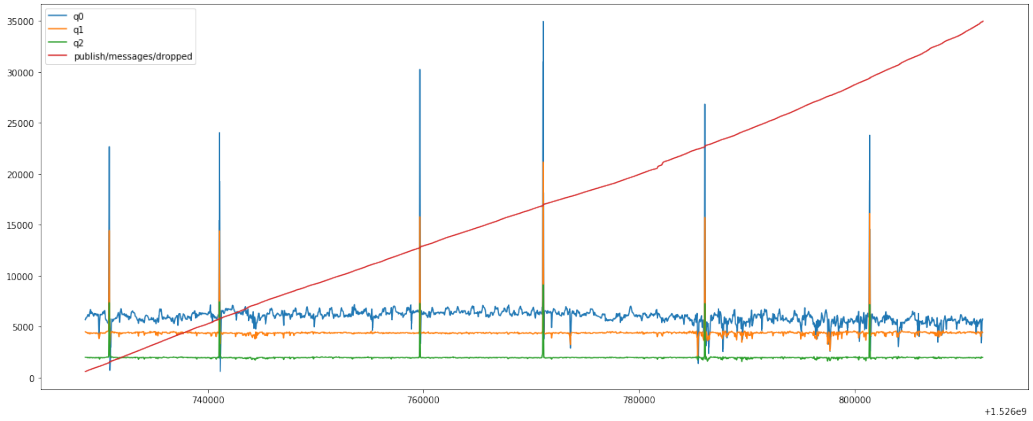


Figure 14: topic: retained